إيجاد أكبر قيمة و أصغر قيمة و الجمع التراكمي :

- دائماً لحساب أكبر رقم من بين مجموعة أرقام :

نفرض أن الرقم الأول هو الكبير max ثم نختبر باقي الأرقام و كلما ظهر رقم أكبر جديد نجعله هو max وهكذا حتى تنتهي مجموعة الأرقام

للرقم الأصغر نقوم بالمثل و لكن كلما ظهر رقم أصغر جديد نجعله min

- عند إجراء عملية الجمع بشكل تراكمي (أي حساب مجموع عدد كبير من الأرقام بشكل متكرر) يجب فرض قيم ابتدائية للمجموع sum و ذلك لأن التعليمة التالية :

sum += x ;

تعني إضافة الرقم x إلى القيمة القديمة للمجموع sum أي هي نفسها التعليمة sum=sum+x;

و بالتالي لا بد من وجود قيمة قديمة للمجموع sum و هذه القيمة القديمة :

١. إما أن تكون صفر بحيث تضاف الأرقام إلى متحول خالي من أي قيمة سابقة
٢. أو أن تكون القيمة الابتدائية للمجموع sum هي أول رقم من الأرقام المراد جمعها بحيث تضاف بقية الأرقام لاحقاً إلى أول رقم

مثال ١ :

اكتب برنامج لقراءة N رقم ثم حساب مجموع هذه الأرقام ومتوسطها وأكبر وأصغر رقم فيها.

```
#include <iostream>
using namespace std;
main( )
{
  int n , x , sum , max , min ;
  cout<<" enter n : " ; cin >> n ; // عدد الأرقام
  cout<<" enter the first number : " ; cin >> x; // أول رقم
  sum = x ; // وضعنا أول رقم في المجموع التراكمي
  min = x ; // فرضنا أول رقم هو الأصغر
  max = x ; // فرضنا أول رقم هو الأكبر
  for (int i = 2 ; i<=n ; i++)  // حلقة للمرور على بقية الأرقام
  {
     cout <<" enter number: " ; cin >> x;  //  إدخال رقم جديد
     sum += x ;  // إضافة x الجديدة للمجموع
     if (x > max) max = x ; //   تغيير قيمة max حسب المقارنة
     if ( x < min) min = x  ; //  تغيير قيمة min حسب المقارنة
  }
  cout << " sum is : " << sum <<"\n";
```

```
  cout << " avg is : " << (float) sum/n <<"\n";
  cout << " max is : " << max <<"\n";
  cout << " min is : " << min <<"\n";
  return 0 ;
}
```

```
enter n : 6
enter the first number : 5
enter number: 0
enter number: 8
enter number: 9
enter number: 4
enter number: 7
sum is : 33
avg is : 5.5
max is : 9
min is : 0
```

التوابع Functions :

من المتعارف عليه أن أفضل طريقة لتطوير وصيانة برنامج كبير تتمثل في عملية تجزئته إلى أجزاء أصغر منه يمكن التحكم فيها بسهولة أكبر من البرنامج الأصلي. وتقدم لغة الـ C++ عدة إمكانيات تساعد في عملية تطوير البرامج الكبيرة و تصميمها وتشغيلها وصيانتها ، حيث نسمي الأجزاء الأساسية لبرنامج بلغة C++ بالتوابع functions التي سنتناولها في هذا الفصل و الصفوف classes التي سندرسها الفصل القادم. وإن معظم برامج لغة الـ C++ يتم عادة كتابتها بالجمع مابين التوابع المكتوبة من قبل المبرمج و التوابع الجاهزة المتوفرة في المكتبة المعيارية للغة الـ C

سوف ندرس في هذا القسم ما يخص التوابع فقط .

تحتوي المكتبة المعيارية للغة C مجموعة غنية من التوابع الخاصة بتنفيذ العمليات الحسابية الرياضية الشائعة ، و العمليات الخاصة بسلاسل الحروف ، بالإضافة إلى عمليات الدخل / الخرج ، و غيرها من العمليات الأخرى المفيدة .

مكتبة التوابع الرياضية :

تسمح توابع المكتبة الرياضية للمبرمج بالقيام بالكثير من الحسابات الرياضية الشائعة. و عند استخدام توابعها يجب القيام بتضمين الملف الرأسي **math.h** داخل نص البرنامج باستخدام التوجيه التالي : **# include < math.h >**

نذكر من هذه التوابع ما يلي :

تعريف التابع	**وصفه**	**مثال**
int abs (int x)	تابع القيمة المطلقة لـ x	abs (-1) =1 , abs (1) =1 abs (0) =0
Double ceil (double x)	تقريب x إلى أصغر عدد صحيح أكبر من x	ceil (5.5) = 6 ceil (- 9.5) = - 9
Double floor (double x)	تقريب x إلى أكبر عدد صحيح أصغر من x	floor (8.3) = 8.0 floor (- 8.3) =- 9
Double fmod (double x , double y)	باقي القسمة الحقيقية(ذو الفاصلة العائمة) لـ x / y	fmod (5.5 , 2 .33)=0.84 fmod (5.5 , 5)=0.5
Double pow (double x , double y)	X مرفوعة للقوة y أي(x^y)	pow(3.0 , 7.0)=2187
Double pow10 (int x)	10 مرفوعة للقوة x	pow10 (3)=1000
Double sin (double x)	جيب الزاوية x (x مقدرة بالراديان)	Sin(90) = 0.893997
Double cos (double x)	تجيب الزاوية x (x مقدرة بالراديان)	cos(90) = - 0.448073
Double tan (double x)	ظل الزاوية x (x مقدرة بالراديان)	tan(90) = -1.9952
Double log (double x)	تابع اللغاريتم الطبيعي لـ x ذو القاعدةe	log(2.8) =1.02962
Double log10 (double x)	تابع اللغاريتم العشري لـ x	log10(2.8) = 0.447158
Double exp (double x)	التابع الأسي e^x	exp(2.5) =12.1825
Double sqrt (double x)	الجذر التربيعي لـ x	sqrt(4) = 2

يجري استدعاء التوابع السابقة بأن نكتب اسم التابع متبوعاً بقوس يساري مفتوح ثم وسطاء التابع (إذا كان أكثر من وسيط يتم الفصل فيما بينهما بواسط فاصلة ,) ثم نضع أخيراً القوس اليميني .

مثال :

```
#include <iostream>
#include<math.h>
using namespace std;
main( )
{       double  x,y;
        cout<<"x="; cin>>x ;
        cout<<"y=";cin>>y;
        cout <<"ceil(x)="<< ceil(x)<<endl ;
        cout<<"fmod(x,y)="<< fmod(x,y) <<endl;
        cout<<"pow( x ,y)=" << pow( x ,y) <<endl;
        cout<<"log( x)="<<log( x)<<endl;
        cout<<"exp( x)="<<exp(x)<<endl;
        cout<<"sqrt(x)="<<sqrt(x)<<endl;
       return 0;
 }
```

```
x=7
y=4
ceil(x)=7
fmod(x,y)=3
pow( x ,y)=2401
log( x)=1.94591
exp( x)=1096.63
sqrt(x)=2.64575
```

تعريف التوابع :

تحتوي كل البرامج التي سبق شرحها على التابع ( ) main الذي نقوم بواسطته استدعاء توابع المكتبة المعيارية وكذلك التوابع المعرفة من قبل المبرمج لتنفيذ مهام البرنامج . ويمكن تحديد الشكل العام لتعريف التوابع كما يلي :

ترويسة التابع // (الوسطاء)اسم التابع نمط الإعادة أو void

بداية جسم التابع // {

; التصريح عن المتحولات المحلية

تعليمات التابع;

; إعادة القيمة في حال لم يكن نمط الإعادة void

نهاية جسم التابع // }

يمكن أن نختار كاسم للتابع أي معرف صحيح **identifier** . في حين أن نمط الإعادة يحدد نمط المعطيات التي يعيدها التابع إلى التابع المستدعي له ويمكن أن نختار أي نمط مثل **.. .. .., int , float , double** ويمكن أن نختار الكلمة **void** كنمط يعيده التابع وذلك للإشارة إلى أن التابع لا يعيد أية قيمة كالتوابع التي تقوم بالطباعة فقط أو التوابع التي تقوم بالقراءة فقط .

أما الوسطاء تتضمن لائحة بالتصريحات عن الوسطاء التي يفصل بينها إشارات الفاصلة '**,**' . حيث يستعين التابع بهذه الوسطاء لتلقي المعلومات التي يجب أن تكون معلومة قبل أن يبدأ التابع بحل المسألة المطلوبة، ويجب تحديد نمط كل وسيط من الوسطاء ضمن القائمة السابقة عند تعريف أي تابع من التوابع .

يتألف جسم البرنامج من مجموعة من التصريحات عن المتحولات المحلية **declarations** **وعدد من التعليمات statements** .

يوجد ثلاث طرق لإعادة التحكم إلى النقطة التي جرى من عندها استدعاء أحد التوابع أي إلى التابع الرئيسي main :

- إذا كان التابع لايعيد أية قيمة **void** فيمكن إعادة التحكم إلى التابع الرئيسي main بكل بساطة وذلك عند الوصول إلى القوس الكبير الذي يحدد نهاية التابع.
- أما إذا كان التابع يعيد نتيجة معينة فإن التعليمة **return متحول ;** تعيد قيمة المتحول إلى التابع الرئيسي وتعيد التحكم له.

ملاحظات:

١. إذا كان التابع لايتلقى أية معلومات فتكون قائمة وسطائه من النمط **void**(الفارغ) .

٢. لا يمكن تعريف تابع ضمن تابع آخر في أي ظرف من الظروف.

٣. إذا لم يحدد نمط التابع فيكون افتراضياً من النمط **int** .

٤. يسبب عدم تحديد نمط النتيجة المعادة من قبل التابع أثناء عملية تعريفه ، خطأً قواعدياً إذا كان نموذج ذلك التابع يحدد نمطاً لتلك النتيجة مختلف عن النمط int .

٥. إن لغة الـ **C++** لا تضمن سوى التوابع **function** ، فكل برنامج جزئي يجب أن يعيد قيمة من نمط معين ، وفي الحالة التي يعيد فيها التابع قيمة من النمط **void** (الفارغ) يتم اختيارها في الحالات التي يقوم فيها البرنامج الجزئي بأعمال معينة لا تنصب على حساب قيمة معينة بذاتها.

٦. تسبب عملية إعادة قيمة لتابع تم التصريح عنه أنه من النمط **void** خطأً قواعدياً.

٧. يؤدي القيام بالتصريح عن الوسطاء من نفس النمط على الشكل التالي ضمن ترويسة التابع **int x , y** بدلاً من الشكل **int x , int y** إلى حدوث خطأً أثناء عملية الترجمة لأنه يجب تحديد نمط كل وسيط من وسطاء التابع ضمن ترويسة التابع .

٨ . تسبب عملية إعادة التصريح عن وسيط من وسطاء التابع موجود ضمن الترويسة على أنه متحول محلي له موجود ضمن جسم التابع ، خطأً قواعدياً.

نماذج التوابع (function prototype) :

يعتبر نموذج التابع من الأمور الهامة التي تخبر المترجم عن نمط المعطيات المعاد من قبل التابع و عن عدد وسطائه و أنماطها و ترتيبها ، حيث يقوم المترجم باستخدامها للتحقق من صحة طريقة استدعاء التابع ، و نموذج التابع هو نفس ترويسة التابع وذلك بعد حذف أسماء الوسطاء منه. الاختلاف الثاني بين ترويسة التابع و نموذج التابع أن نموذج التابع لا بد أن يكون متبوعاً بفاصلة منقوطة.

أي على الشكل :

void نمط الإعادة أو اسم التابع (أنماط الوسطاء);

فمثلاً يبين النموذج التالي :

int max (int , int) ;

أن التابع max يأخذ وسيطين من النمط int ويعيد نتيجة من نفس النمط int .

مثال٢ : أكتب برنامج يتضمن التوابع التالية :

- لحساب مربع عدد
- التابع الرئيسي main الذي يستدعي التابع السابق

```
#include <iostream>
```

```
using namespace std;
double sq(double);     //  نموذج التابع

main( )
{
  double x ;
 cout<<"enter the number:";  cin>> x ;
 cout<<"square is: "<<sq(x)<<endl ;      //  استدعاء التابع
 return 0;
}
// تابع حساب مربع عدد //
double sq (double  y)
{
 return y*y;
 }
```

```
enter the number:8
square is: 64
```

مثال٣ :

أكتب برنامج يتضمن التوابع التالية :

- لحساب القيمة الكبرى بين عددين
- التابع الرئيسي main الذي يستدعي التابع السابق

```
#include <iostream>
using namespace std;
int max( int  , int ) ; //  نموذج التابع
main( )
{
   int x , y , z ;
   cout<<"Enter tow numbers: ";cin >> x >> y ;
   z = max( x , y ) ;     //  استدعاء التابع
   cout<<"maximum is: "<< z<<endl ;
   return 0;
}

int max( int x , int y )
{
   if (x > y )
```

```
Enter tow numbers: 7 9
maximum is: 9
```

```
        return x ;
    else
        return y ;
}
```

مثال٤ :

أكتب برنامج يتضمن التوابع التالية :

- تابع لطباعة الأعداد من 0 إلى x
- التابع الرئيسي main الذي يستدعي التابع السابق

```
#include <iostream>
using namespace std;
void print ( int ) ;    // نموذج التابع
main( )
{
  int x ;
  cout<<"Enter a number : ";cin >>x;
  print(x);    // استدعاء التابع
  return 0;
}
void print (int x)
{
   for (int i=0;i <=x ; i ++)
     cout<< i <<endl ;
}
```

```
Enter a number : 7
0
1
2
3
4
5
6
7
```